Primjena modela za linearnu regresiju na danom skupu podataka

Novak, Lucija

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:513644

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-02-16**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Lucija Novak

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

doc. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Lucija Novak

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradila samostalno pritom koristeći znanje stečeno tijekom studija uz navedenu literaturu.

Zahvaljujem se mentoru doc. dr. sc. Tomislavu Stipančiću na svoj pruženoj pomoći prilikom izrade i pisanja rada te uloženom vremenu i trudu.

Posebno se zahvaljujem svojoj obitelji i prijateljima koji su mi sve studentske dane činili lakšima.

Lucija Novak

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Lucija Novak** JMBAG: **0035217179**

Naslov rada na hrvatskom jeziku: **Primjena modela za linearnu regresiju na danom skupu podataka**

Naslov rada na engleskom jeziku: **Application of a linear regression model for the given data set**

Opis zadatka:

Matematička metoda linearne regresije jedna je od najučinkovitijih metoda za brojne primjene umjetne inteligencije i znanosti o podacima. U odnosu na druge metode strojnog učenja krasi je jednostavnost, djelotvornost i interpretabilnost.
U radu je potrebno razviti računalni model temeljen na metodi linearne regresije koristeći odgovarajući skup podataka.

Rad treba sadržavati:

- opis matematičkog značenja metode linearne regresije,
- objašnjenje i implementaciju metode gradijentnog spuštanja (eng. Gradient Descent) kao algoritma za minimiziranje funkcije gubitka koja je temeljena na korekciji trenutnih težinskih faktora računalnog modela,
- odabir parametara za učenje te rada kreiranog modela (npr. stopa učenja, broj prolazaka kroz model i sl.),
- interpretaciju rada modela te kritički osvrt na dobivene rezultate.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
9. 5. 2022.	**2. rok (izvanredni):** 6. 7. 2022. **3. rok:** 22. 9. 2022.	**2. rok (izvanredni):** 8. 7. 2022. **3. rok:** 26. 9. – 30. 9. 2022.
Zadatak zadao:		Predsjednik Povjerenstva:
Doc. dr. sc. Tomislav Stipančić		Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
a	/	nagib pravca
b	/	odsječak pravca na y-osi
b_i	/	parametar
c	/	odsjek
db	/	parcijalna derivacija funkcije gubitka s obzirom na odstupanje
dc	/	parcijalna derivacija funkcije gubitka s obzirom na odsjek
dm	/	parcijalna derivacija funkcije gubitka s obzirom na nagib
dw	/	parcijalna derivacija funkcije gubitka s obzirom na težinu
dx	/	promjena x
dy	/	promjena y
m	/	nagib
n	/	količina
w	/	težina
x	/	varijabla, neovisan član
y	/	varijabla, ovisan član

Grčka slova

Oznaka	Jedinica	Opis
α	/	stopa učenja

POPIS FORMULA

SAŽETAK

Tema ovog rada je primjena modela linearne regresije na danom skupu podataka. Model i njegova implementacija razvija se u programskom jeziku Python. Učitavanje biblioteka, provjera i podjela podataka i njihovo testiranje odrađuju glavni zadatak treniranja takvog modela. Model linearne regresije ima za cilj predviđati podatke, a vizualiziranjem krajnjeg ishoda procjenjuje se točnost i uspješnost. Svemu tome prethodi i objašnjenje teorijske osnove potrebne za razumijevanje.

Ključne riječi: linearna regresija, Python, gradijentno spuštanje, parametar, funkcija

SUMMARY

The topic of this paper is the application of the linear regression model for the given data set. The model and its implementation are developed in the Python programming language. Importing librarires, verifying and splitting data and testing them perform the main task of training such a model. A linear regression model aims to predict data, and by visualizing the end result, accuracy and success are assessed. All this is preceded by an explanation of the theoretical basis necessary for understandig.

Key words: linear regression, Python, gradient desccent, parameter, function

1. UVOD

Otkad je ljudi postoji nagon za otkrivanjem, tumačenjem i pojašnjavanjem činjenica. Istraživanje dovodi do svakakvih saznanja, ali i nepobitnih činjenica. Provjereni dokazi i iskustvo temelj su tzv. empirijskih istraživanja, a regresijska analiza jedna je od korištenih metoda u modernijim empirijskim istraživanjima.

Linearna je regresija bila prvi tip regresijske analize te će detaljnije biti opisana u ovom radu. Zbog prisutnosti linearne ovisnosti o nepoznatim parametrima lakše se modelira nego modeli s nelinearnom ovisnošću o parametrima. Podešavanje modela linearne regresije česta je uz pomoć minimiziranja funkcije gubitaka. Potrebno je prikupiti podatke ciljane teme i ustanoviti jednadžbu modela linearne regresije te treningom postići što točnije i preciznije predviđanje rješenja. [1][2]

2. TEORIJSKA OSNOVA RADA

2.1.Strojno učenje

2.1.1. O strojnom učenju

Strojno je učenje (enlg. *machine learning*) grana umjetne inteligencija koja se bavi oblikovanjem algoritama koji svoju učinkovitost poboljšavaju na temelju empirijskih podataka. Strojno učenje jedno je od danas najaktivnijih i najuzbudljivijih područja računarske znanosti, ponajviše zbog brojnih mogućnosti primjene koje se protežu od raspoznavanja uzoraka i dubinske analize podataka do robotike, računalnog vida, bioinformatike i računalne lingvistike. [3]

2.1.2. Način rada strojnog učenja

Strojno učenje je polje koje se tiče projektiranja i razvoja algoritama i tehnika koje omogućuju računalu da uči. Za strojno učenje ključni su podaci i model strojnog učenja. Model strojnog učenja „hrani“ se podacima i ono pokušava učiti iz njih te pronaći uzorak za iste situacije. Kad su mu dani novi podaci mogu se probati kreirati i nove vrijednosti.

Ako se želi postići da model strojnog učenja nalazi razinu krvnog šećera osobe. Uzet će se primjereni model strojnog učenja i trenirati ga dajući mu puno podataka povezanih s dijabetesom. Kad je model naučen i upoznat s tim podacima onda mu se da novi skup podataka i on može naći razinu šećera u krvi zbog poznatih parametara. To je način na koji svaki model strojnog učenja radi.

2.2.Model linearne regresije

2.2.1. Način rada modela linearne regresije

Shvaćanje rada modela linearne regresije može se dobro objasniti uz pomoć primjera.

Ako postoji skup podataka koji sadrži vrijednosti plaća za određenu profesiju i iskustvo rada u toj profesiji izraženu u godinama i postavlja se pitanje kolika je plaća osobe koja radi tri godine onda nam je jasno da će s porastom iskustva rasti i plaća, tj. događa se linearni porast [Tablica 1.1].

Pokušava se pronaći što povezuje te podatke. To ne mora nužno biti pravac jer vrijednosti mogu odstupati. Linearna regresija je to što postavlja te vrijednosti da pašu pravcu. To je metoda pomoću koje se mogu naći jednadžbe linije i slične vrijednosti.

Tablica 2.1. Primjer usporedbe plaće i godina iskustva

x – neovisan član	1	2	3	4	5
y – ovisan član	5	7	9	11	13

Za dane podatke vrijedi klasična jednadžba pravca (2.1.).

$$y = ax + b \tag{2.1.}$$

gdje je:

a – nagib pravca

b – odsječak na y-osi.

Nagib pravca nam zapravo govori kako se mijenja y s obzirom na x (2.2.).

$$a = \frac{dy}{dx} \tag{2.2.}$$

gdje je:

dy – promjena y

dx – promjena x.

$$a = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.3.}$$

$$a = \frac{7-5}{2-1} = 2 \tag{2.4.}$$

Vrijednost nagiba jednaka je 2. To znači da ako se vrijednosti x-osi mijenjaju za 1, onda se vrijednosti y-osi mijenjaju za 2.

Sada nam je poznata vrijednost nagiba, ali još moramo saznati vrijednost odsječka na y-osi:

$$y = 2x + b \tag{2.5.}$$

Vrijednost nagiba saznajemo uvrštavanjem bilo kojeg para vrijednosti. npr, x=1, y=5.

$$5 = 2 \times 1 + b \tag{2.6.}$$

$$b = 3$$

Zaključak:

$$y = 2x + 3 \tag{2.7.}$$

je funkcija koja uspostavlja odnos između x i y. Za danu vrijednost x, može se naći y.

To je strojno učenje. Pokušava se pronaći linija da paše podacima i odgovarajući y. Kad je linearni model, kao u ovom primjeru, priča se o linearnoj regresiji. Model se mijenja, ali koncept ostaje isti. Kod dvije vrste podataka (x i y) model uči upravo iz tih podataka. Tako treniran model može se koristiti za ostvarivanje novih pretpostavki.

2.2.2. Dvije ili više varijabli

Kod višestruke linearne regresije postoji jedan ovisan član (y), a dva ili više neovisnih (x). Mora se postaviti model za predviđanje vrijednosti jedne ovisne varijable o dvije ili više neovisne. Kao što je u prvom primjeru plaća ovisila o iskustvu, sad će npr. ovisiti o iskustvu i edukaciji. Tada se mora koristiti 3D model te se više ne govori o pravcu, nego o ravnini.

Jednostavna linearna regresija (2.8.)

$$y = b_0 + b_1 x_1 \tag{2.8.}$$

Višestruka linearna regresija (2.9.)

$$y = b_0 + b_1 x_1 + \ b_2 x_2 + \cdots + \ b_n x_n \tag{2.9.}$$

2.2.3. Prednosti i nedostaci

Prednosti:

- Jednostavno za izvršiti. Implementacija je jednostavna jer ne treba puno toga. Najbitniji je nagib i odsječak na y-osi (a i b).
- Dobro izvođenje s podacima koji imaju linearnu vezu. Ako se x poveća, poveća se i y. Za takvu ovisnost lako se koristi linearna regresija, a ne neki kompliciraniji model.

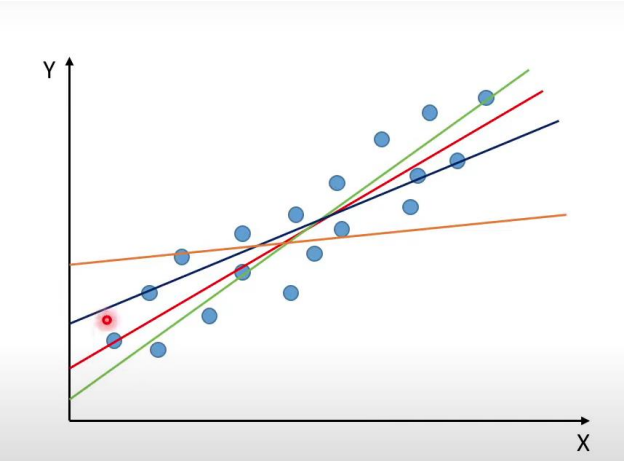
Nedostaci:

- Nije pogodno kad podaci imaju nelinearnu vezu, npr. sinusoida.
- Problem nedovoljne opremljenosti. Kada model ne može naći uzorak prisutan u podacima i povezati ih.
- Osjetljiv na ekstreme.

3. IZVEDBA ZADATKA

3.1.Matematičko razumijevanje

Nakon shvaćanja intuicije iza modela linearne regresije i kako naći parametre nagiba i odsjeka, potrebno je shvatiti matematiku linearne regresije i kako se model linearne regresije uklapa u skup podataka.



Slika 3.1. Različiti načini povezivanja podataka

Kad postoji skup linearnih podataka u kojima kada raste x, raste i y kroz njih se može provući bezbroj linija koji ih na neki način povezuju [Slika 3.1.]. Svaka linija ima svoj nagib i odsjek, ali cilj je pronaći onu koja najbolje povezuje te podatke.

3.2.Funkcija gubitka

Funkcija je gubitka (engl. *loss function*) funkcija koja uspoređuje ciljane i predviđene izlazne vrijednosti te mjeri koliko je procijenjena vrijednost daleko od prave vrijednosti. Korisna je prilikom odredbe koji parametar ima bolju izvedbu i koji je parametar bolji. Vrijednosti su dvije: predviđena i prava. Predviđena vrijednost predviđena je našim modelom linearne regresije, a prava su podaci koje imamo. Funkcija gubitka govori koji je od četiri pravaca sa slike 3.1. najbolji. (3.1.)

$$GUBITAK = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (3.1.)$$

Gdje je:

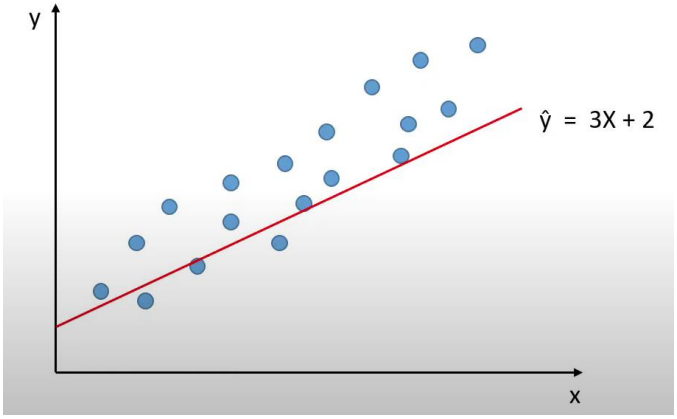
y_i – prava vrijednost

$\hat{y}_i$ – procijenjena vrijednost.

Na primjeru slučajno odabranih parametara i vrijednosti objasnit će se funkcija gubitka. Parametri su:

$$a = 3, b = 2$$

$$\hat{y} = 3x + 2 \tag{3.2.}$$



Slika 3.2. Odabrani pravac

Tablica 3.1. Primjeri za funkciju gubitka

x	y	Ŷ
2	10	8
3	14	11
4	18	14
5	22	17
6	26	20

Ako se x i ŷ „plotaju“ dobit će se pravac, a ako isto to napravimo sa x i y dobit će se točkice prikazane na slici iznad [slika 3.2.].

$$GUBITAK = \frac{1}{5} \times [(10-8)^2 + (14-11)^2 + (18-14)^2 + (22-17)^2 + (26-20)^2] \quad (3.3)$$

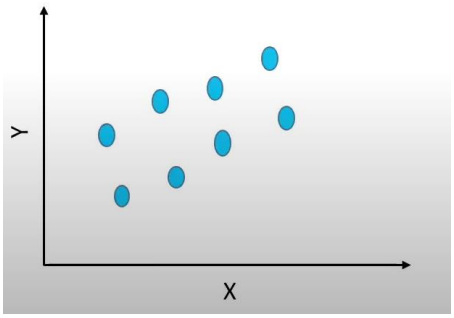
$$GUBITAK = \frac{1}{5} \times (4 + 9 + 16 + 25 + 36) = 18 \quad (3.4.)$$

Gubitak treba biti što manji, to znači da su previđene i stvarne vrijednosti jako blizu. Niska vrijednost gubitka ukazuje na visoku točnost. Kad model „shvati“ da je gubitak visok, kao u ovom slučaju, reducirat će gubitak iteracijom.

3.3.Gradijentno spuštanje za linearnu regresiju

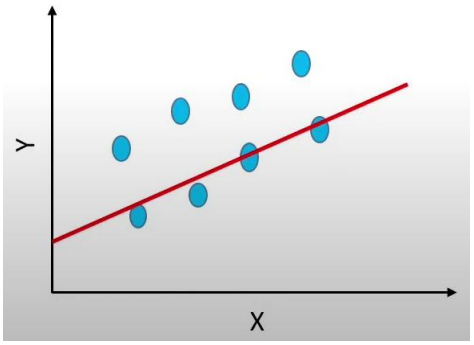
3.3.1. Optimizacija modela

Gradijentno je spuštanje (eng. *gradient descent*) tehnika optimizacije modela koja se koristi kako bi najbolji parametri za model bili nađeni, gubitak smanjio i time omogućio modelu preciznije predviđanje.



Slika 3.3. Podaci

Na slici je prikazano osam parova podataka (x,y) [Slika 3.3.]. S obzirom na te podatke može se pronaći puno pravaca koji bi na određene načine zadovoljavale predviđanja.



Slika 3.4.Y_1

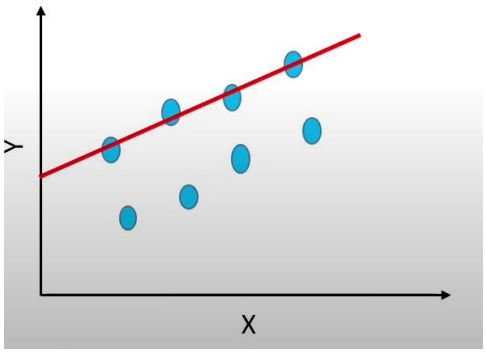
Na slici je prikazana pronađena y_1 funkcija (3.5.) [Slika 3.4.]:

$$y_1 = m_1 x + c_1 \tag{3.5.}$$

S obzirom na tu funkciju, četiri para podataka bliže su crvenoj liniji, a četiri para su dalje.

Drugi mogući slučaj je za funkciju y_2 koja glasi (3.6) [Slika 3.5.]:

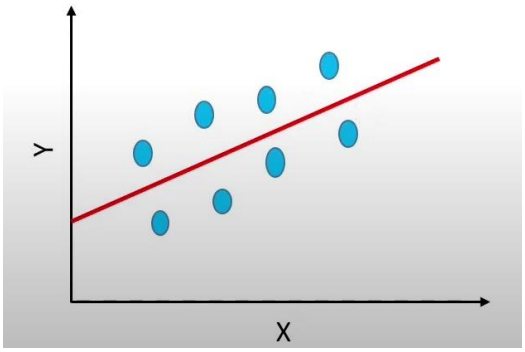
$$y_2 = m_2 x + c_2 \tag{3.6}$$



Slika 3.5. Y_2

Također, moguća je i funkcija y_3 (3.7.) [Slika 3.6.]:

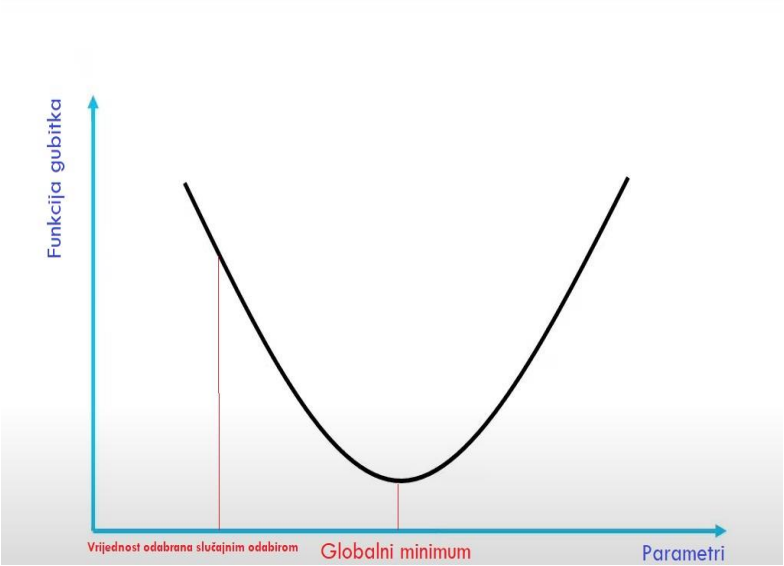
$$y_3 = m_3 x + c_3 \tag{3.7.}$$



Slika 3.6. Y_3

Bez obzira što linija y_3 ne prolazi kroz čak ni jedan podatak, m_3 i c_3 parametri prepoznati su kao najbolji jer su najbliži svim podacima.

Kad se parametri i odgovarajuća funkcija gubitka „plotaju“ dobiva se krivulja u-oblika. Traži se globalni minimum u kojem će funkcija gubitka biti najmanja i imat će najbolje optimizirane parametre [Slika 3.7.]. Dolazak do toga zahtjeva iteracije. Model linearne regresije će se kretati po u-krivlji dok ne dođe do najboljih parametara.



Slika 3.7. Globalni minimum

3.3.2. Gradijentno spuštanje

Gradijentno spuštanje je optimizacijski algoritam korišten za minimiziranje funkcije gubitka u različitim algoritmima strojnog učenja. Korišten je za ažuriranje parametara modela učenja. To se provodi kroz sljedeće bitne formule koje će se implementirati i u Python-u. (3.8.) (3.9)

$$m = m - \alpha \times D_m \tag{3.8.}$$

$$c = c - \alpha \times D_c \tag{3.9.}$$

gdje su:

m – nagib

c – odsjek

α – stopa učenja

D_m – parcijalna derivacija funkcije gubitka s obzirom na nagib

D_c – parcijalna derivacija funkcije gubitka s obzirom na odsjek.

Stopa učenja je vrijednost mijenjanja parametara, a parcijalne derivaciju po m i c parametru označavaju koliko se promijeni funkcija gubitka kada promijenimo m ili c.

Parcijalna derivacija s obzirom na nagib može se još objasniti kao diferencijal funkcije troška (engl. *cost function*). Funkcija troška i funkcija gubitka su jako slične i često se i koriste upravo kao sinonimi. Razlika između funkcije troška i funkcije gubitka je u tome što funkcija gubitka izračunava grešku za jedan primjer, dok funkcija troška izračunava prosjek funkcija gubitka svih primjera.

Parcijalna derivacija s obzirom na nagib izračunava se na sljedeći način. Prvo se parcijalno derivira funkcija troška s obzirom na nagib. Funkcija troška mjeri udaljenost između stvarnih i predviđenih vrijednosti.

$$D_m = \frac{\partial(Funkcija\ troška)}{\partial m} = \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=0}^{n}\left(y_i - \ y_{pred}\right)^2\right) \quad (3.10.)$$

Dolazi do zamijene predviđenog y (y_{pred}) sa $mx_i + c$ jer je to jednadžba predviđene funkcije (3.11.). Na slici 3.2. y predstavlja plave točke, a y_{pred} predviđene vrijednosti.

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}(y_i - \ (mx_i + c))^2\right) \quad (3.11.)$$

Nakon te implementacije dolazi do kvadriranja razlike:

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}(y_i^2 + \ m^2x_i^2 + \ c^2 + 2mx_ic - 2y_imx_i - 2y_ic\right) \quad (3.12.)$$

Dobivena kvadrirana razlika diferencira se s obzirom na m (3.13.).

$$= \frac{-2}{n}\sum_{i=0}^{n} x_i\left(y_i - \ (mx_i + c)\right) \quad (3.13.)$$

Nakon diferencijacije opet dolazi do zamijene (3.14.).

$$= \frac{-2}{n} \sum_{i=0}^{n} x_i \left(y_i - \ y_{i,pred}\right) \tag{3.14.}$$

Parcijalna derivacija funkcije troška s obzirom na odsječak na y-osi provodi se slično. Funkcija troška parcijalno se derivira ovoga puta s obzirom na c varijablu (3.15).

$$D_c = \frac{\partial (Funkcija\ troška)}{\partial c} = \frac{\partial}{\partial c} \left(\frac{1}{n} \sum_{i=0}^{n} \left(y_i - \ y_{pred}\right)^2\right) \tag{3.15.}$$

Dolazi do zamijene predviđenog y (y_{pred})sa $mx_i + c$ jer je to jednadžba predviđene funkcije (3.16.)

$$= \frac{1}{n} \frac{\partial}{\partial c} \left(\sum_{i=0}^{n} (y_i - \ (mx_i + c))^2\right) \tag{3.16.}$$

Nakon te implementacije dolazi do kvadriranja razlike (3.17.).

$$= \frac{1}{n} \frac{\partial}{\partial c} \left(\sum_{i=0}^{n} (y_i^2 + \ m^2 x_i^2 + \ c^2 + 2mx_i c - 2y_i mx_i - 2y_i c\right) \tag{3.17.}$$

Dobivena kvadrirana razlika diferencira se s obzirom na c (3.18).

$$= \frac{-2}{n} \sum_{i=0}^{n} \left(y_i - \ (mx_i + c)\right) \tag{3.18.}$$

Nakon diferencijacije opet dolazi do zamijene (3.19.).

$$= \frac{-2}{n} \sum_{i=0}^{n} \left(y_i - y_{i,pred} \right) \quad (3.19.)$$

3.4.Izgradnja linearne regresije u Pythonu

Jednadžba modela linearne regresije glasi

$$y = wx + b \quad (3.20.)$$

gdje su:

y – ovisna varijabla

x – neovisna varijabla

w – težina (eng. *weight*)

b – odstupanje (eng. *bias*)

Gradijentno spuštanje ovog modela linearne regresije dano je jednadžbama 3.21. i 3.22.

$$w = w - \alpha \times dw \quad (3.21.)$$

$$b = b - \alpha \times db \quad (3.22.)$$

gdje su:

α – stopa učenja

dw – parcijalna derivacija funkcije gubitka s obzirom na težinu

db – parcijalna derivacija funkcije gubitka s obzirom na odstupanje.

U daljnje obrađivanom uzorku skupa podataka Podaci o plaći (*salary_data*) ima 30 točaka podataka [Tablica 3.2.]. Sadrži godine iskustva i mjesečne plaće. Ti podaci se trebaju iskoristiti na način da se model linearne regresije smjesti u te podatke i radi nove pretpostavke. U jednadžbi ovisna varijabla predstavljat će mjesečnu plaću, a neovisna varijabla godine iskustva.

Tablica 3.2. Podaci o plaći

	Iskustvo [godina]	*Plaća [INR]*
1.	1,1	37 731
2.	1,3	39 343
3.	1,5	39 891
4.	2	43 525
5.	2,2	46 205
6.	2,9	54 445
7.	3	55 794
8.	3,2	56 642
9.	3,2	56 957
10.	3,7	57 081
11.	3,9	57 189
12.	4	60 150
13.	4	61 111
14.	4,1	63 218
15.	4,5	64 445
16.	4,9	66 029
17.	5,1	67 938
18.	5,3	81 363
19.	5,9	83 088
20.	6	91 738
21.	6,8	93 940
22.	7,1	98 273
23.	7,9	101 302
24.	8,2	105 582
25.	8,7	109 431

26.	9	112 635
27.	9,5	113 812
28.	9,6	116 969
29.	10,3	121 872
30.	10,5	122 391

3.4.1. Kako naći najbolje parametre?

Funkciju gubitka treba obraditi za različite parametre i odrediti koji skup parametara dovodi do najmanje funkcije gubitka ili troška. Ažuriranje podataka provodi se jednadžbama (3.21.) i (3.22.). Stopa učenja je parametar podešavanja u optimizacijskom algoritmu koji određuje veličinu koraka u svakoj iteraciji prilikom traženja minimalne funkcije gubitka. Parcijalne derivacije funkcije gubitka određuju koliko se funkcija gubitka promjeni prilikom promjene parametara (3.23.) (3.24).

$$dw = \frac{-2}{n}\sum_{i=0}^{n} x_i\left(y_i - y_{i,pred}\right) \tag{3.23.}$$

$$db = \frac{-2}{n}\sum_{i=0}^{n}\left(y_i - y_{i,pred}\right) \tag{3.24.}$$

Jednadžba (3.23.) ponaša se kao jednadžba (3.14.) iz prošlog poglavlja, a jednadžba (3.24.) kao (3.19.).

Jednadžba (3.20.) je jednadžba linije, jednadžbe (3.21.) i (3.22.) predstavljaju ažuriranje podataka našeg modela, a (3.23.) i (3.24.) stopu učenja.

3.4.2. Python

Python je moćan i lako razumljiv programski jezik. Programi pisani njime obično su kraći, a za njihovo pisanje utroši se manje vremena. Pokazao se i kao jezik koji može zamijeniti BASIC i Pascal u učenju osnova programiranja. Podržava različite stilove programiranja, od strukturalnog do objektno-orijentiranog. Osim toga, podržava integriranje dijelova programskog koda napisanih u drugim jezicima. Ciklus uređivanja, testiranja i otklanjanja pogrešaka je brz i na visokoj razini. Pregledavaju se lokalne i globalne varijable te se prolazi kroz kod redak po redak. Program za ispravljanje grešaka napisan je u samom Pythonu. Python je besplatan, izdaje se pod Open Source licencom. [4] [5]

3.4.3. NumPy

Numpy (Numerical Python) je Python biblioteka (engl. *library*) otvorenog koda koja s koristi u gotovo svim područjima znanosti i inženjerstva. Ta biblioteka je standard za rad s numeričkim podacima u Pythonu. NumPy API se intenzivno koristi u Pandas, SciPy, Matplotlib, scikit-learn, scikit-image i većini drugih podatkovnih i znanstvenih Python paketa. Biblioteka NumPy sadrži višedimenzionalne strukture podataka nizova i matrica.

NumPy se može koristiti za izvođenje širokog spektra matematičkih operacija na nizovima. Pythonu dodaje moćne podatkovne strukture koje jamče učinkovite izračune s nizovima i matricama i nudi ogromnu biblioteku matematičkih funkcija visoke razine koje rade s tim matricama i nizovima. [6]

3.4.4. Izgradnja modela linearne regresije

Izgradnja modela započinje učitavanjem numPy biblioteke koja nam je potrebna za rad s nizovima. Linearnu regresiju uvodimo kao klasu (eng. *class*) koja je u Pythonu korištena za izradu novog lokalnog prostora gdje su definirani svi njeni atributi. Atributi mogu biti podaci ili funkcije. U klasi se izrađuju šablone u koje se mogu uključiti višestruke funkcije koje se kod treniranja modela uvode. Klasa stvara novi lokalni prostor gdje su definirani svi njeni atributi. Atributi mogu biti podaci ili funkcije.U ovom slučaju potrebna je funkcija uvođenja (engl. *initiation function*),

funkcija treniranja (engl. *fit function*), funkcija ažuriranja (engl. *updation function*) te funkcije predviđanja (engl. *predict function*). Funkcije u model linearne regresije uključujemo pozivom. [Slika 3.8.]

```
import numpy as np

class Linear_Regression():
    def __init__():
    def fit():
    def update_weights(s):
    def predict():
```

Slika 3.8. Početak izgradnje modela

Funkcija uvođenja uvodi, tj. inicira podatke, zatim funkcija treniranja trenira model linearne regresije sa danim podacima. Nakon ažuriranja podataka model sam predviđa tražene vrijednosti. Kao što definicije funkcija počinju s ključnom riječi *def* u Pythonu, tako i definicije klase počinju ključnom riječi *class*.

3.4.4.1.Funkcija uvođenja

Funkcija uvođenja sadržavat će tri parametra. Prvi i najbitniji je parametar *self*. Taj parametar u Pythonu koristi se za sve instance u klasi i predstavlja instancu samog objekta. U većini jezika ovaj parametar prolazi skriveno dok u Pythonu se mora izričito navesti. Njime se lako može pristupiti svim instancama definiranim unutar klase, uključujući njezine metode i atribute.

Nakon nje inicira se stopa učenja i iteracija, tzv. hiperparametri (eng. *hyper parameters*). To su parametri za koje će biti potrebno ručno unesti iznos. Stopa učenja će tu djelovati kao magnituda promjene među parametrima. [Slika 3.9.]

```
def __init__(self, learning_rate, no_of_iterations):
    self.learning_rate = learning_rate
    self.no_of_iterations = no_of_iterations
```

Slika 3.9. Definiranje funkcije uvođenja

3.4.4.2.Funkcija treniranja

Za razliku od funkcije uvođenja funkcija treniranja, uz parametar *self*, sadrži parametre težine i odstupanja. Oni se ne postavljaju ručno već su određeni modelom linearne regresije, tzv. parametri modela (engl. *model parameters*).

X shape, tj. u obliku niza postavljaju se m i n vrijednosti. Gdje će m označavati primjere za treniranje, a n značajke. U ovom primjeru samo je jedna značajka, tj. to je x vrijednost.

Unutar funkcije kreira se matrica ili niz sa značajkama ili kolumnama. Sa n brojem kolumna u kojem sve vrijednosti sadrže nule. Podaci mogu sadržavati različit broj značajki i svaka značajka ima težinu povezanu s tom značajkom. Težina će biti matrica u kojoj model ima samo jednu vrijednost odstupanja. Vrijednosti x i y bit će dane.

Zbog uvođenja algoritma gradijentnog snižavanja za optimizaciju koristi se *for* petlja. Pomoću nje model radi iteraciju te uspoređuje funkciju gubitka i svaki put ažurira podatke. Zbog toga se mora kreirati funkcija ažuriranja. [Slika 3.10.]

```
def fit(self, X, Y):
        self.m, self.n = X.shape

        self.w = np.zeros(self.n)
        self.b = 0
        self.X = X
        self.Y = Y

        for i in range(self.no_of_iterations):
            self.update_weights()
```

Slika 3.10. Definiranje funkcije treniranja

3.4.4.3.Funkcija ažuriranja

Funkcija ažuriranja sadrži samo parametar *self* te u jednadžbu (3.20.) daje vrijednost x i pronalazi vrijednosti y i sve vrijednosti predviđene modelom sprema u niz. Nakon toga uspoređuje predviđene y vrijednosti s pravima. [Slika 3.11.]

```
def update_weights(self):
        Y_prediction = self.predict(self.X)

        dw = - (2 * (self.X.T).dot(self.Y - Y_prediction)) / self.m
        db = - 2 * np.sum(self.Y - Y_prediction)/self.m

        self.w = self.w - self.learning_rate*dw
        self.b = self.b - self.learning_rate*db
```

Slika 3.11. Definiranje funkcije ažuriranja

3.4.4.4.Funkcija predviđanja

U funkciji predviđanja uz parametar *self* nalazi se i parametar x. Razlog tome je kada se da vrijednost godina iskustva može se izračunati plaća, tj. daje se samo jedan parametar. U ovom slučaju to je upravo parametar x.

Kad se modelu da ta vrijednost probat će uključiti tu vrijednost u formulu (3.20.) te naći y. [Slika 3.11.]

```
def predict(self, X):

    return X.dot(self.w) + self.b
```

Slika 3.11. Definiranje funkcije predviđanja

Kad se završi broj iteracija dolazi se do točke zvane konvergencija. To je točka nakon koje model ne može bolje izvoditi i u toj točki je funkcija gubitka jako mala, tj. globalni minimum.

3.4.5. Provedba linearne regresije u Python-u

U sljedećem koraku podaci se implementiraju u model linearne regresije. Nakon što se model istrenira na tim podacima, daju mu se novi podaci i vrijednosti te je vidljiva vrijednost plaće koju model može predvidjeti.

Tijek rada modela linearne regresije može se podijeliti u šest koraka:

1) U prvom koraku postavlja se stopa učenja i broj iteracija. Iniciraju se slučajno odabrane vrijednosti težine i odstupanja

2) Izgradnja jednadžbe linearne regresije
3) Pronalazak predviđene y vrijednosti za dani x uz odgovarajuću težinu i odstupanje
4) Provjera funkcije gubitka za te vrijednosti parametara
5) Ažuriranje vrijednosti parametara koristeći gradijentno snižavanje (nove vrijednosti težine i odstupanja)
6) Ponavljanje koraka 3, 4, 5 dok ne dođe do minimalne funkcije gubitka.

Onda se dobije najbolji model, tj. najbolje vrijednosti težine i odstupanja zbog pronalaska minimalne funkcije gubitka.

3.4.5.1.Panda

Panda je Python biblioteka korištena kao brz, moćan, fleksibilan i jednostavan alat za analizu podataka. Izgrađena je na bazi Python programskog jezika. Biblioteka panda često čisti neuredne skupove podataka i čini ih čitljivima i relevantnima. [7]

3.4.5.2.Scikit-learn (Sklearn)

Scikit-learn (Sklearn) je najkorisnija i najsnažnija biblioteka za strojno učenje u Pythonu. Pruža izbor učinkovitih alata za strojno učenje i statističko modeliranje uključujući klasifikaciju, regresiju, grupiranje i smanjenje dimenzionalnosti putem konzistentnog sučelja u Pythonu. Ova biblioteka, koja je uglavnom napisana u Pythonu, izgrađena je na NumPy, SciPy i Matplotlib. [8]

3.4.5.3.Matplotlib

Matplotlib je biblioteka niske razine za crtanje grafova u Pythonu. Služi kao program za vizualizaciju. Otvorenog je koda i uglavnom je napisan u Pythonu. Nekoliko segmenata napisano je u C-u, Objective-C i Javascriptu za komatiblinost s platformom. [9]

3.4.5.4.Upotreba modela linearne regresije za predviđanje

Početak radnog tijeka započinje učitavanjem svih gore navedenih biblioteka u Python skriptu. [Slika 3.13.]

```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Slika 3.13. Učitavanje biblioteka

3.4.5.5.Prethodna obrada podataka

Daljnje procesiranje podataka ukazat će na to postoje li neke vrijednosti koje nedostaju u podacima. Učitavaju se podaci u csv formi (engl. *comma separated values*) gdje su sve vrijednosti razdvojene zarezom u panda podatkovni okvir.

Uz pomoć *head* funkcije prikazuje se prvih pet redaka podataka. [Slika 3.14.]

```
First five rows

   YearsExperience  Salary
0              1.1   39343
1              1.3   46205
2              1.5   37731
3              2.0   43525
4              2.2   39891
```

Slika 3.14. Prvih pet redova podataka

Tail funkcija izlistava zadnjih 5 redaka podataka [Slika 3.15]. Imamo 30 parova podataka, ali prvi počinje sa nulom te je zadnji podatak označen rednim brojem 29.

```
Last five rows

    YearsExperience  Salary
25              9.0  105582
26              9.5  116969
27              9.6  112635
28             10.3  122391
29             10.5  121872
```

Slika 3.15. Zadnjih pet redova podataka

Bitno je znati koliko redova ima skup podataka koji se koristi te nedostaje li neki podatak. [Slika 3.16.]

```
(30, 2)
YearsExperience    0
Salary             0
```

Slika 3.16. Broj redova i stupaca i vrijednosti koje nedostaju

3.4.5.6.Razdvajanje svojstvenih i ciljanih vrijednosti (x i y)

Svojstvene vrijednosti se pohranjuju kao x vrijednosti, a ciljane vrijednosti kao y vrijednosti. Kod svojstvenih vrijednosti potrebno je maknuti stupac koji daje vrijednosti plaća. Jedan stupac će biti pokazan, a drugi neće. Vrijednost godine iskustava pohranjuje se u x vrijednosti. Za y vrijednosti prikazivat će se samo zadnji stupac u kojem su pohranjene plaće.[Slika 3.17]

```
[[ 1.1]
 [ 1.3]
 [ 1.5]
 [ 2. ]
 [ 2.2]
 [ 2.9]
 [ 3. ]
 [ 3.2]
 [ 3.2]
 [ 3.7]
 [ 3.9]
 [ 4. ]
 [ 9.5]
 [ 9.6]
 [10.3]
 [10.5]]
[ 39343  46205  37731  43525  39891  56642  60150  54445  64445  57189
  63218  55794  56957  57081  61111  67938  66029  83088  81363  93940
  91738  98273 101302 113812 109431 105582 116969 112635 122391 121872]
```

Slika 3.17. Razdvajanje svojstvenih i ciljanih vrijednosti

3.4.5.7.Razdvajanje podataka na podatke za obuku i podatke za testiranje

Radi razdvajanja podataka već je na početku učitana *train_test_split* funkcija koja će napraviti četiri niza. U ispitnom setu bit će testirano oko 30% podataka, a ostali podaci (oko 70%) bit će iskorišteni za sami trening. Tako razdvojeni podaci spremni su za treniranje modela.[Slika 3.18.]

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.33, random_state = 2)
```

Slika 3.18. Razdvajanje podataka na podatke za obuku i podatke za testiranje

3.4.5.8.Treniranje modela linearne regresije

Model linearne regresije trenira se sa x i y vrijednostima za treniranje i onda se testira na predviđanje plaće. Klasa linearne regresije učitava se te dobiva parametre kao što su stopa učenja i broj iteracije čije vrijednosti se moraju dati.

Tu kreće treniranje modela linearne regresije koji je sadržan u funkciji treniranja i bitan je za učenje modela. Unutar toga bit će provedeni svih šest koraka rada modela linearne regresije. X i y vrijednosti bit će vrijednosti treniranja modela koji su već podijeljeni u *test_train_split* funkciji.

Ispis parametara težine i odstupanja provest će se na način da će vrijednost odstupanja biti cijela vrijednost, a težina numPy niz [Slika 3.19.]. Te slijedi predviđanje plaće.

```
weight =  9514.400999035135
bias =  23697.406507136307
```

Slika 3.19. Ispis parametara

Tražena funkcija po kojoj će sada model tražiti plaću je

$$y = 9514x + 23\,697$$

$$plaća = 9514(godine\ iskustva) + 23\,697$$

3.4.5.9.Predviđanje vrijednosti plaće sa podacima za testiranje

Danom vrijednosti x model će predvidjeti y s obzirom na jednadžbu sa slike. Sve predviđene vrijednosti stavljaju se u niz. [Slika 3.20.]

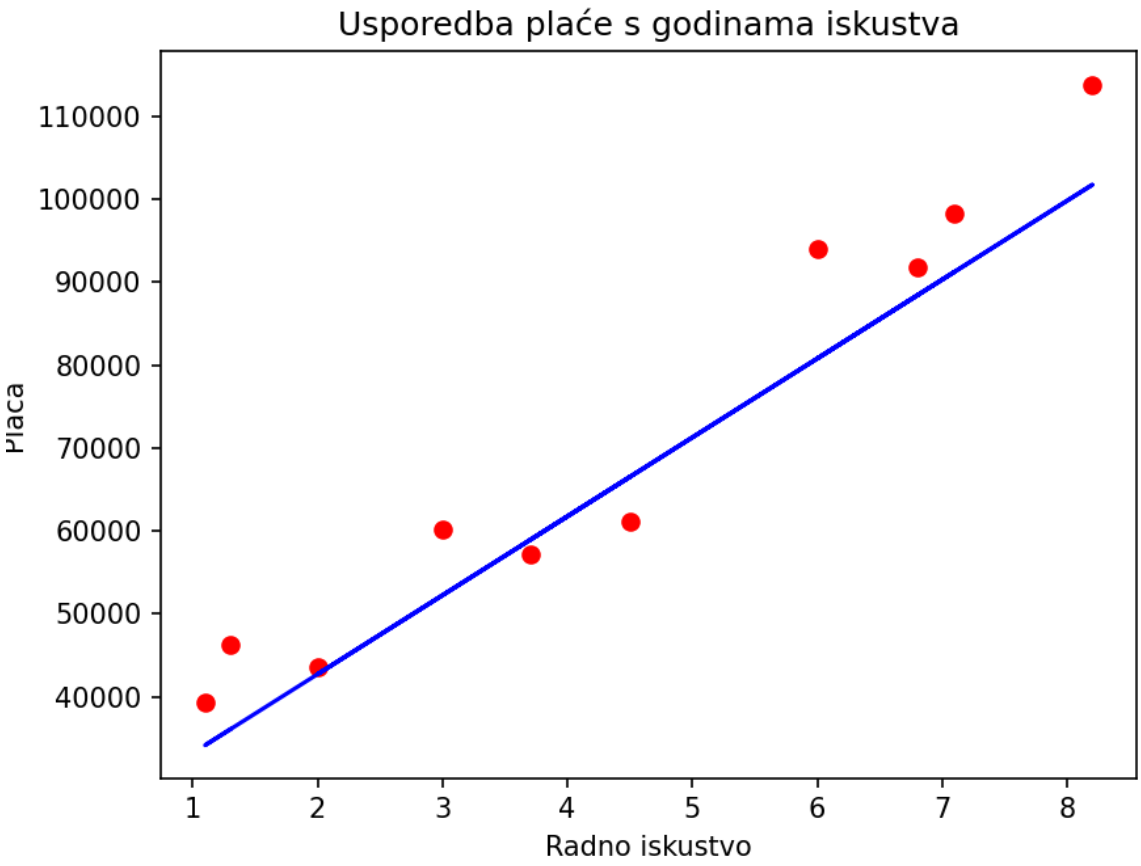
```
[ 36066.12780588  34163.24760607  66512.21100279  58900.69020357
  91249.65360029  80783.81250135 101715.49469922  52240.60950424
  42726.20850521  88395.33330058]
```

3.20. Ispis predviđenih vrijednosti

3.4.5.10. Vizualiziranje predviđenih i stvarnih vrijednosti

Kada se priča o predviđanju klasa misli se na klasifikaciju kod koje se koristi ocjena točnosti kao metrika procjene. Regresija je zato predviđanje vrijednosti i za nju se često koristi srednja apsolutna greška, srednja kvadratna greška itd. Također, jako dobro rješenje za procjenu modela je vizualiziranje prave i predviđene vrijednosti.

Za vizualiziranje je potreban graf u kojem je model linearne regresije „plotan“ s x vrijednostima za testiranje gdje plava linija predstavlja vrijednosti predviđene modelom, a crvene točke predstavljaju prave vrijednosti [Slika 3.21.]. Prava vrijednost prisutna u podacima je Y_test i uspoređuje se sa test_data_predcition nizom. Ako su točke i linija blizu na traženom grafu znači da model dobro predviđa vrijednosti. Nije moguće napraviti model koji prolazi kroz sve točke.



Slika 3.21. Vizualiziranje predviđenih i stvarnih vrijednosti

4. ZAKLJUČAK

U ovom radu prikazana je izgradnja modela linearne regresije i implementacija modela u programskom jeziku Python. Model je treniran da predviđa vrijednosti koje na kraju, uspoređujući ih s pravim vrijednostima, vizualizira radi procjene točnosti.

Ovaj model linearne regresije može se ocijeniti kao model koji dobro predviđa. Gradijentno spuštanje je dovelo do zadovoljavajuće točke globalnog minimuma.

LITERATURA

[1] https://eu.questionpro.de/bs/empirijsko-istra%C5%BEivanje/#empirische_forschung_definition dostupno dana 19. 8.2022.

[2] https://www.hrstud.unizg.hr/images/50040570/OSUD-P10.pdf dostupno dana 19. 8.2022.

[3] https://www.fer.unizg.hr/predmet/struce1 dostupno dana 24 8.2022.

[4] https://www.python.org/ dostupno dana 23. 8.2022.

[5] https://e.udzbenik.hr/priPy/files/upute_install.pdf dostupno dana 23. 8.2022.

[6] https://numpy.org/devdocs/user/absolute_beginners.html dostupno dana 23. 8.2022.

[7] https://pandas.pydata.org/ dostupno dana 24. 8.2022.

[8] https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm dostupno dana 24. 8.2022.

[9] https://www.w3schools.com/python/matplotlib_intro.asp dostupno dana 24. 8.2022.

PRILOZI

I.Python kod

```python
#importing numPy library
import numpy as np

#linear regression as an object
class Linear_Regression():
    def __init__(self, learning_rate, no_of_iterations):

        self.learning_rate = learning_rate
        self.no_of_iterations = no_of_iterations


    def fit(self, X, Y):
        self.m, self.n = X.shape

        #iniciranje težine i odsjeka
        self.w = np.zeros(self.n)
        self.b = 0
        self.X = X
        self.Y = Y

        #uključivanje gradijentnog spuštanja radi optimizacije
        for i in range(self.no_of_iterations):
            self.update_weights()
```

```python

    def update_weights(self):

        Y_prediction = self.predict(self.X)

        #izračunavanje gradijenta
        dw = - (2 * (self.X.T).dot(self.Y - Y_prediction)) / self.m
        db = - 2 * np.sum(self.Y - Y_prediction)/self.m

        #ažuriranje
        self.w = self.w - self.learning_rate*dw
        self.b = self.b - self.learning_rate*db

    def predict(self, X):
        return X.dot(self.w) + self.b #y=wx+b

```

```
43  #uvođenje knjižnica
44  import pandas as pd
45  from sklearn.model_selection import train_test_split
46  import matplotlib.pyplot as plt
47
48  #učitavanje podataka kao .csv dokument
49  salary_data = pd.read_csv(r'C:\Users\novak\Desktop\Završni\salary_data.csv')
50
51  #ispis prvih 5 stupaca
52  print("First five rows\n")
53  t=salary_data.head()
54  print(t)
55
56  #ispis zadnjih pet stupaca
57  print("Last five rows\n")
58  c=salary_data.tail()
59  print(c)
60
61  #broj stupaca i redova
62  e=salary_data.shape
63  print(e)
64
65  #provjera nedostalih vrijednosti
66  r=salary_data.isnull().sum()
67  print(r)
68
```

```
69  #podjela x i y
70  X = salary_data.iloc[:,:-1].values
71  Y = salary_data.iloc[:,1].values
72
73  print (X)
74
75  print (Y)
76
77  #podjela na podatke za testiranje i podatke za treniranje
78  X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.33, random_state = 2)
79
80  #treniranje modela linearne regresije
81  model = Linear_Regression(learning_rate = 0.02, no_of_iterations=1000)
82
83  model.fit(X_train, Y_train)
84
85  #ispis vrijednosti težine i odsjeka
86  print('weight = ', model.w[0])
87  print('bias = ', model.b)
88
89  #predviđanje plaće
90  test_data_prediction = model.predict(X_test)
91
92  print(test_data_prediction)
93
```

```
94   #graf
95   plt.scatter(X_test, Y_test, color = 'red')
96   plt.plot(X_test, test_data_prediction, color = 'blue')
97   plt.xlabel('Radno iskustvo')
98   plt.ylabel('Plaća')
99   plt.title('Usporedba plaće s godinama iskustva')
100  plt.show()
```